UWAGA!
Treść niniejszego dokumentu powstała na podstawie cyklu artykułów pt. „Mikrokontrolery? To takie proste” zamieszczonych w czasopiśmie Elektronika dla Wszystkich.

Asembler - język maszynowy procesora

Operacje przemieszczania danych

1. Instrukcja ***MOV***

Instrukcja służąca do przekazywania danych pomiędzy rejestrami procesora i pamięcią wewnętrzną.
W zależności od tego co i gdzie „przenosimy”, polecenie to może mieć kilkanaście różnych postaci w zależności od zastosowanych argumentów. Jednocześnie warto wiedzieć, że polecenie to w praktyce nie powoduje dosłownego ‘przemieszczenia’ danej, lub zawartości rejestru, ale jej skopiowanie za źródła do miejsca przeznaczenia. Przypomina to polecenie KOPIUJ - WKLEJ.
Ogólnie instrukcję MOV można zapisać jako:

MOV <d> <s>
gdzie <d> jest miejscem przeznaczenia (ang. destination), a <s> źródłem pobrania danej (ang. surce). W wyniku wykonania instrukcji MOV zawartość źródła <s> zostaje umieszczona (skopiowana)
w obiekcie przeznaczenia <d>. W prawie wszystkich przypadkach (oprócz jednego) argumentami instrukcji MOV są wyrażenia 8-bitowe: rejestry, dane adresy pośrednie itp. Jedynie załadowanie 16-bitowego wskaźnika adresu DPTR wymaga odpowiedniego 16-bitowego argumentu.
Poniżej opisane są wszystkie możliwe przypadki użycia instrukcji MOV.

MOV A,Rn
- do akumulatora zostaje załadowana zawartość rejestru Rn
- kod: 1 1 1 0 1 n2 n1 n0 gdzie n2...n0 wskazują na R0...7 stąd: E8-EFh

MOV A,adres
- do akumulatora zastaje załadowana zawartość komórki wewnętrznej pamięci RAM o adresie: „adres”
- kod: 11100101 E5h

MOV A,@Ri
- do akumulatora zostaje załadowana zawartość komórki wew. RAM, której adres znajduje się w rejestrze R0 (i=0), lub R1 (i=1)
- kod: 1110011i stąd: E6h, E7h

MOV A,#dana
- instrukcja załadowania 8-bitowej liczby „dana” do akumulatora
- kod: 01110100 74h

MOV Rn,A
- do rejestru Rn (R0...R7) zostaje załadowana zawartość akumulatora
- kod: 1 1 1 1 1 n2 n1 n0 gdzie n2..n0 wskazują na R0...R7 stąd: F8h-FFh

MOV Rn,adres
- do rejestru Rn (R0...R7) zastaje załadowana zawartość komórki o adresie „adres”
- kod: 1 0 1 0 1 n2 n1 n0 gdzie n2...n0 wskazują na R0...R7 stąd: A8h-AFh

MOV Rn,#dana
- do rejestru Rn (R0...R7) zastaje wpisana 8-bitowa liczba
- kod: 0 1 1 1 1 n2 n1 n0 gdzie n2...n0 wskazują na R0...R7 stąd: 78h-7Fh

MOV adres,A
- do komórki o adresie „adres” zostaje wpisana zawartość akumulatora
- kod: 11110101 F5h

MOV adres,Rn
- do komórki o adresie „adres” zostaje wpisana zawartość rejestru Rn (R0...R7)
- kod: 1 0 0 0 1 n2 n1 n0 stąd: 88h-8Fh

MOV adres1,adres2
- przepisanie zawartości komórki o adresie „adres2” do komórki o adresie „adres1”
- kod: 10000101

MOV adres,@Ri
- do komórki o adresie „adres” zostaje wpisana zawartość komórki której adres znajduje się w rejestrze R0 (i=0) lub R1 (i=1)
- kod: 1000011i gdzie i=0 lub i=1 stąd: 86h, 87h

MOV adres, #dana
- do komórki o adresie „adres” zostaje wpisana 8-bitowa liczba
- kod: 01110101 75h

MOV @Ri,A
- do komórki o adresie znajdującym się w rejestrze R0 (i=1) lub R1 (i=1) zostaw wpisana zawartość akumulatora
- kod: 1111011i gdzie i=0 lub i=1 stąd: F6h, F7h

MOV @Ri, adres
- do komórki o adresie znajdującym się w rejestrze R0 (i=0) lub (i=1) zostaje wpisana zawartość komórki o adresie „adres”
- kod: 1010011i gdzie i=0 lub i=1

MOV @Ri, #dana
- do komórki o adresie znajdującym się w rejestrze R0 (i=0) lub R1 (i=1) zostaje wpisana podana 8-bitowa dana
- kod: 0111011i gdzie i=0 lub i=1 stąd 76h, 77h

MOV DPTR, #dana16

- instrukcja załadowania 16-bitowego, bezwzględnego adresu do wskaźnika danych DPTR. „Dana 16” jest liczbą 16-bitową, czyli z zakresu 0...FFFFh
- kod: 10010000

2. Instrukcja ***MOVC***

Podobną do instrukcji **MOV** jest ***MOVC.*** Służy ona także do przemieszczania danych z tym, że przemieszczanie dotyczy tylko pobierania danych (bajtów) znajdujących się w kodzie programu, czyli w wew. lub zewnętrznej pamięci programu procesora. W praktyce instrukcje tę wykorzystuje się do pobierania danych stałych (np. tablic). Innym często spotykanym przypadkiem jest generowanie standardowych komunikatów np. na wyświetlaczach LCD. Ponieważ takie komunikaty są z reguły niezmienna, w praktyce programista umieszcza je w kodzie programu (pamięci stałej).

MOVC A,@A+DPTR

- do akumulatora zostaje załadowana dana z pamięci programu spod adresu będącego sumą bieżącej wartości wskaźnika danych DPTR i zawartości akumulatora. Najpierw procesor tworzy 16-bitowy adres poprzez dodanie DPTR i A, potem pobiera spod tego adresu daną (bajt kodu programu) i umieszcza ją w akumulatorze
- kod: 10010011 93h

MOVC A,@A+PC

- do akumulatora zostaje załadowana dana z pamięci programu spod adresu będącego sumą wartości: licznika rozkazów PC (następnej po tej instrukcji) i zawartości akumulatora. W praktyce przy wykonaniu tej instrukcji adres pobrania jest równy sumie zawartości akumulatora oraz wartości licznika rozkazów - będzie to adres następnej po MOVC instrukcji
- kod: 10000011 83h

3. Instrukcja ***MOVX***

Służy ona do przesyłania danej pomiędzy akumulatorem a zewnętrzną pamięcią danych. Wykonanie tej instrukcji uaktywnia sygnały /RD („/” oznacza negację) (przy odczycie z zewnętrznej pamięci) lub /WR (przy zapisie) procesora - piny P3.7 i P3.6. Dodatkowo porty P0 i P2 pełnią wtedy rolę magistrali systemowej dzięki której wystawiany jest adres oraz przekazywana dana do zewnętrznej pamięci danych.

Zewnętrzną pamięć danych można zaadresować w dwojaki sposób. Pierwszą metodą jest użycie pełnego 16-bitowego adresu. W takim przypadku procesor odczytując lub zapisując daną w tej pamięci (właśnie dzięki instrukcji MOVX) młodszą część adresu zatrzaskuje w wewnętrznym lath'u (np. 74573), starszą zaś wystawia na port P2.

Często jednak używana kostka pamięci SRAM jest mniejszej pojemności i większość linii adresowych starszego bajtu (adresu) nie jest wykorzystana do sterowania pamięcią. W takim przypadku możliwe jest adresowanie pamięci za pomocą tzw. „stronicowania”. W takim trybie adresowania procesor przy obsłudze zewnętrznej pamięci danych wystawie tylko młodszą część adresu (A0...A7), natomiast port P2 nie jest modyfikowany, co pozwala użytkownikowi na pełną kontrolę sposobu i kierunku ustawienia jego końcówek - a więc jest metodą na maksymalne wykorzystanie cechy „jednoukładowości” procesora.

I tak dwa wspomniane tryby adresowania zewnętrznej RAM mają swoje odbicie w liście instrukcji z wykorzystaniem rozkazu MOVX, oto one.

Tryb pełnego adresu (16-bitowego)

MOVX A, @DPTR

- do akumulatora zostaje załadowana dana z zewnętrznej pamięci danych (odczyt z zewnętrznej pamięci danych) spod adresu który jest w DPTR
- kod: 11100000 E0h

MOVX @DPTR, A

- do komórki zewnętrznej pamięci danych o podanym w DPTR adresie zostaje przesłana zawartość akumulatora - jest to więc operacja zapisu do zewnętrznej pamięci danych
- kod: 11110000 F0h

Tryb stronicowania (niepełnego adresu)

MOVX A, @Ri

- do akumulatora zostaje przesłana zawartość komórki w obszarze zewn. pamięci danych spod adresu znajdującego się w rejestrze R0 (i=0) lun R1 (i=1): adres 8-bitowy
- kod: 1110001i gdzie Ri=R0 lub R1 stąd: E2, E3
- przykład:
  niech w układzie z procesorem 8951 (jest to procesor kompatybilny z 8051) pracującym z zew. pamięcią programu znajduje się zewnętrzna pamięć danych w postaci kostki SRAM 2kB - typ 6116.
  Linie adresowe A0...A7 tej pamięci są dołączone do zatrzasku młodszej części adresu szyny procesora. Trzy starsze linie A8...A10 są dołączone np. do pinów P2.0, P2.1, P2.2 procesora, pozostałe końcówki portu P2 (P2.3...P2.7) są wykorzystywane np. do sterowania przekaźnikami jakiegoś urządzenia zewnętrznego. Aby odczytać daną z tej pamięci np. spod adresu 24h na stronie pierwszej (strony liczone od 0 do 7, bo 2kB / 256=8 stron) należy wykonać następujące instrukcje:

```
CLR  P2.2     ;wyzerowanie linii         ;adresowej A10
CLR  P2.1     ;wyzerowanie linii         ;adresowej A9
SETB  P2.0    ;ustawienie linii
              ;adresowej A8
               (strona 1)
MOV  R1,#24h   ;załadowanie
              ;adresu komórki do
              wskaźnika
MOVX  A, @R1    ;i przesłanie jej
               ;zawartości do akumulatora
...
```

Przy takim zaadresowaniu pamięci nie uległy modyfikacji piny P2.3...P2.7 portu P2 procesora, co w wielu przypadkach jest wręcz niezbędne. Można by oczywiście zaadresować tę pamięć za pomocą instrukcji MOVX A,@DPTR (podając wtedy adres MOV DPTR, #0124h), ale wtedy zniszczeniu uległy by stany pozostałych, nie dołączonych do pamięci końcówek portu P2.

MOVX @Ri,A
- do obszaru zewnętrznej pamięci danych o adresie znajdującym się w rejestrze Ri zostaje przesłana zawartość akumulatora. Innymi słowy jest to zapis do zewnętrznej pamięci danych
- kod: 1111001i   gdzie i=0,1

4. Instrukcje przesyłania wymiany danych ze stosem

PUSH adres
- w wyniku tej operacji zawartość wskaźnika stosu SP jest zwiększana o 1, po czym na wierzchołek stosu (adresie w wew. RAM wskazywanym przez SP) zostaje zapisana zawartość komórki z wew. RAM o podanym adresie bezpośrednim „adres". Wykonywana jest operacja przesyłania na stos
- kod: 11000000   C0h
- przykład:
  PUSH ACC   ;akumulator na stos
  PUSH B   ;rejestr B na stos
  PUSH 20h   ;zawartość kom. 20h na stos
- UWAGA!!!
  przesyłając akumulator na stos piszemy **PUSH ACC**

POP adres
- dana znajdująca się pod adresem w wew. RAM określonym w SP zostaje wpisana do komórki o podanym adresie bezpośrednim „adres". Następnie wskaźnik stosu SP zostaje zmniejszony o 1 *(w niektórych typach procesorów jest odwrotnie, tzn. przesyłając daną na stos SP jest zmniejszany o 1, a przy odczycie zwiększany o 1).* Wykonywana jest operacja zdjęcia ze stosu.
- kod: 11010000   D0h

5. Dodatkowe instrukcje przemieszczania danych

XCH A,adres
- zawartość akumulatora zostaje wymieniona z zawartością komórki w wew. RAM o podanym adresie bezpośrednim
- kod: 11000101   C5

XCH A,@Ri
- zawartość akumulatora zostaje wymieniona z zawartością komórki w wew. RAM o adresie znajdującym się w rejestrze R0 lub R1
- kod: 1100011i   i=0,1

XCHD A, @Ri
- młodszy półbajt (bity 0-3) akumulatora zostaje wymieniony z młodszym półbajtem komórki w wew. RAM o adresie zawartym w R0
- kod: 1101011i   i=0,1

Operacje na bitach

Procesor 8051 i mu pochodne zawiera bardzo pomocny zestaw instrukcji do wykonywania na pojedynczych bitach. Dzięki temu możliwe jest wykonanie wielu często niezbędnych operacji. Większość rejestrów specjalnych SFR procesora posiada możliwość bezpośredniego adresowania ich bitów. Akumulator np. składa się z 8-miu adresowanych bitów Acc.7... Acc.0 Aby zatem np. ustawić wybrane bity tego rejestru nie trzeba modyfikować całości a jedynie wyzerować lub ustawić wybrany bit. Dla przykładu prześledźmy sytuację gdy chcemy wyzerować bit 4 akumulatora bez integrowania w pozostałe. Można wykonać te zadanie dwojako:

- poprzez instrukcję iloczynu logicznego:

  ANL A, #11101111b

- lub poprzez instrukcję działającą na pojedynczym bicie:

  CLR Acc.4

Instrukcje operujące na bitach nabierają szczególnie praktycznego znaczenia przy badaniu stanu końcówek (portów) mikroprocesora lub przy ich sterowaniu (ustawianiu na nich poziomów logicznych niskich lub wysokich oraz przy ustawianiu w stan wysokiej impedancji - tzw. „trzeci stan”). Możemy wówczas operować na pojedynczym bicie (pojedynczej końcówce) nie naruszając przy okazji innych do których mogą być przyłączone urządzenia.

CLR C

- wyzerowany zostaje znacznik (bit w rejestrze PSW - stanu) przeniesienia C
- kod: 11000011 C3h

SETB C

- ustawiona zostaje flaga przeniesienia
- kod: 11010011 D3h

CLR bit

- wyzerowany zostaje bit którego adres podany jest bezpośrednio
- kod: 11000010 C2h

SETB bit

- ustawiony zostaje bit którego adres podany jest bezpośrednio
- kod: 11010010 D2h

CPL C

- flaga C zostaje zanegowana
- kod: 10110011 B3h

CPL bit

- zanegowany zostaje bit którego adres podany jest bezpośrednio
- kod: 10110010 B2h

ANL C,bit

- iloczyn logiczny znacznika C i bitu o adresie podanym jako bezpośredni
- w wyniku tej operacji zostaje wykonany iloczyn logiczny flagi przeniesienia C oraz bitu o adresie „bit”, a wynik zostaje umieszczony w C
- kod: 10000010 82h

ANL C, /bit („/" oznacza negację)

- wykonany zostaje iloczyn logiczny flagi przeniesienia C oraz zanegowanego bitu o adresie „bit", a wynik zostaje umieszczony w C
- kod: 10110000 B0h

ORL C, bit

- suma logiczna flagi przeniesienia C oraz bitu o adresie „bit", a wynik zostaje umieszczony w C
- kod: 01110010 72h

ORL C, /bit

- suma logiczna flagi C i zanegowanego bitu o adresie „bit", a wynik zostaje umieszczony w C
- kod: 10100000 A0h

MOV C,bit

- zawartość bitu o podanym adresie „bit" zostaje przepisana do znacznika przeniesienia C
- kod: 10100010 A2h

MOV bit,C

- zawartość znacznika C zostaje przepisana do bitu o adresie „bit"
- kod: 10010010 92h

dwa powyższe rozkazy są wykorzystywane przy chcemy przenieść zawartość jakiegoś bitu (o podanym adresie) do innego o innym adresie. Do wykonania tego niezbędny jest znacznik C. Wykonujemy to następująco:

```
MOV C, bit1         ;najpierw bit1 do C
MOV  bit2, C        ;a potem z C do bit 2
```

Błędem jest natomiast wykonanie instrukcji:
MOV bit2, bit1
Taki rozkaz nie istnieje!!!

Operacje arytmetyczne

Instrukcja ADD

- do wartości przechowywanej w akumulatorze dodawany jest wskazany argument, a wynik zostaje wpisany do akumulatora
- znaczniki: C, AC i OV

ADD A,Rn

- do akumulatora dodawana jest zawartość rejestru Rn
- kod: 0 0 1 0 1 n2 n1 n0 n2...n0 - R0-R7 stąd: 28h-2Fh

ADD A,adres

- do akumulatora dodawana jest zawartość komórki wew. RAM o adresie „adres"
- kod: 00100101 25h

ADD A, @Ri
- do akumulatora dodawana jest zawartość komórki wew. RAM o adresie wskazywanym przez rejestr Ri
- kod: 0010011i i=0,1 stąd: 26h, 27h

ADD A, #dana
- do akumulatora dodawany jest argument stały (8-bitowa liczba)
- kod: 00100100 24h

Instrukcja **ADDC**
- do wartości przechowywanej w akumulatorze dodawany jest wskazany argument oraz zawartość znacznika przeniesienia C, a wynik zostaje wpisany do akumulatora
- znaczniki: C, AC, OV

ADDC A,Rn
- do akumulatora dodawana jest zawartość rejestru Rn oraz C
- kod: 0 0 1 1 1 n2 n1 n0 n2...n0 - R0-R7 stąd: 38h-3Fh

ADDC A,adres
- do akumulatora dodawana jest zawartość komórki wew. RAM o adresie „adres", oraz znacznik przeniesienia C
- kod: 00110101 35h

ADDC A, @Ri
- do akumulatora dodawana jest zawartość komórki wew. RAM o adresie wskazywanym przez rejestr Ri, oraz znacznik przeniesienia C
- kod: 0011011i i=1,0 stąd: 36h, 37h

ADDC A,dana
- do akumulatora dodawany jest argument stały (8-bitowa liczba), oraz wskaźnik przeniesieni C
- kod:00110100 34h

Instrukcja ***SUBB***
od wartości przechowywanej w akumulatorze odejmowany jest wskazany argument oraz zawartość znacznika C, a wynik zostaje wpisany do akumulatora
- znaczniki: C, AC, OV

SUBB A, Rn
- od A odejmowana jest zawartość rejestru Rn oraz C
- kod: 1 0 0 1 1 n2 n1 n0 n2..n0 - R0 - R7

SUBB A,adres
- od A odejmowana jest zawartość komórki w wew. RAM o adresie „adres", oraz C
- kod: 10010101 95h

SUBB A, @Ri

- od A odejmowana jest zawartość komórki w wew. RAM o adresie wskazywanym przez rejestr Ri (R0 lub R1) oraz C
- kod: 1001011i i=1,0 stąd: 96h, 97h

SUBB A, #dana
- od A odejmowany jest argument stały (8-bit. liczba) oraz C
- kod: 10010100 94h

Instrukcja **INC**
Do wskazanego argumentu dodawana jest jedynka. Znaczniki nie ulegają zmianie

INC A
- do A dodawana jest jedynka
- kod: 00000100 04h

INC Rn
- do zawartości rejestru Rn dodawana jest jedynka
- kod: 0 0 0 0 1 n2 n1 n0 n2...n0 - R0 - R7 stąd: 08h-0Fh

INC adres
- do zawartości komórki o adresie „adres” dodawana jest jedynka
- kod: 00000101 05h

INC @Ri
- do zawartości komórki o adresie wskazywanym przez Ri dodawana jest jedynka
- kod: 0000011i i=0,1 stąd: 06h, 07h

INC DPTR
- do 16 bitowego wskaźnika danych złożonego z rejestrów SFR: DPH (bardziej znaczący bajt) i DPL (mniej znaczący bajt) dodawana jest jedynka
- kod: 10100011 A3h

Instrukcja ***DEC***
Od wskazanego argumentu odejmowana jest jedynka. Znaczniki pozostają bez zmian

DEC A
- od A odejmowana jest jedynka
- kod: 00010100 14h

DEC Rn
- od zawartości rejestru Rn odejmowana jest jedynka
- kod: 0 0 0 1 1 n2 n1 n0 n2...n0 - R0-R7 stąd 18h - 1Fh

DEC adres
- od zawartości komórki o adresie „adres” odejmowana jest jedynka
- kod: 00010101 15h

DEC @Ri

- od zawartości komórki której adres podany jest w rejestrze Ri odejmowana jest jedynka
- kod: 0001011i   i=0,1   stąd: 17h,17h

Instrukcja ***MUL AB***
- pomnóż
- 8-bitowa liczba bez znaku znajdująca się w akumulatorze jest mnożona przez 8-bitową liczbę bez znaku z rejestru B. 16-bitowy wynik wpisywany jest do rejestrów B i A (bardziej znaczący bajt do B, mniej znaczący bajt do A)
- znaczniki: *jeśli wynik mnożenia jest większy od 255 to ustawiony jest znacznik OV, w przeciwnym razie OV jest wyzerowany, znacznik C jest zerowany*
- kod: 10100100   A4h
- zapis:
  MUL AB

Instrukcja ***DIV AB***
- podziel
- 8-bitowa liczba bez znaku, znajdująca się w akumulatorze jest dzielona przez 8-bitową liczbę z rejestru B. Część całkowita ilorazu wpisywana jest do akumulatora, a reszta do rejestru B. *W przypadku gdy dzielnik jest równy 0 (B=0) to po wykonaniu operacji zawartość akumulatora i rejestru B jest nieokreślona oraz dodatkowo ustawiony zostaje znacznik OV*
- znaczniki: C=0   OV=0 (zerowane)
- kod: 10000100   84h
- zapis:
  DIV AB

Instrukcja ***DA A***
- wykonywana jest korekcja dziesiętna wyniku dodawania. Operacja ta sprowadza wynik do postaci dwóch cyfr dziesiętnych w kodzie BCD, jeżeli argumenty były w kodzie BCD. Rozkaz ten powinien być używany jedynie w połączeniu z rozkazem dodawania (ADD, ADDC). Także inkrementacja powinna odbywać się poprzez instrukcje ADD A,#1, a nie INC A bowiem w tym drugim przypadku nie są ustawiane znaczniki C i AC, tak więc nie może być wykonana korekcja dziesiętna. Korekcja polega na tym, że w przypadku kiedy po wykonanej na akumulatorze operacji dodawania (ADD, ADDC) zawartość jego bitów 3...0 jest większa od 9 lub jest ustawiony znacznik AC, to do wartości akumulatora dodawana jest liczba 6. Po tym jeżeli okaże się że zawartość bitów 7...4 jest większa od 9 lub jest ustawiony znacznik C to do tych bitów dodawana jest także 6. Jeżeli podczas tej ostatniej operacji wystąpiło przeniesienie to do znacznika wpisywana jest 1, w przeciwnym wypadku stan znacznika C nie zmienia się
- znaczniki: C, OV
- kod: 11010100   D4h

Operacje logiczne

Instrukcja ***ANL***

- wykonywany jest iloczyn logiczny AND (mnożenie bitów bit po bicie) wskazanych w instrukcji dwóch argumentów. Wynik operacji jest wpisywany do argumentu pierwszego instrukcji
- znaczniki: nie zmieniają się

ANL A,Rn

- wymnożona logicznie zostaje zawartość akumulatora i rejestru Rn, a wynik wpisywany jest do akumulatora
- kod: 0 1 0 1 1 n2 n1 n0   gdzie n2...n0 wskazują na R0...R7 stąd: 58h-5fh

ANL A,adres

- wymnożona zostaje logicznie zawartość akumulatora i komórki o adresie „adres”, wynik zostaje umieszczony w A
- kod: 01010101   55h

ANL A,@Ri

- wymnożona logicznie zostaje zawartość akumulatora i komórki wew. RAM o adresie wskazywanym przez rejestr Ri, wynik zostaje umieszczony w A
- kod: 0101011i   i=0,1 stąd: 56h,57h

ANL A,#dana

- wymnożona logicznie zostaje zawartość akumulatora przez argument stały (8-bitowa liczba), wynik zostaje umieszczony w A
- kod: 01010100   54h

ANL adres, A

- wymnożona logicznie zostaje zawartość komórki pamięci o adresie „adres” i akumulatora, a wynik zostaje umieszczony w komórce o adresie „adres”
- kod: 01010010   52h

ANL adres, #dana

- wymnożona logicznie zostaje zawartość komórki o adresie „adres” przez argument stały (8-bitowa liczba), wynik zostaje umieszczony w komórce o adresie „adres”
- kod: 01010011   53h

Instrukcja ***ORL***

- wykonywana jest suma logiczna OR (dodawanie bitów bit po bicie) wskazanych w instrukcji dwóch argumentów. Wynik operacji jest wpisywany do argumentu pierwszego instrukcji
- znaczniki: nie zmieniają się

ORL A, Rn

- dodana logicznie zostaje zawartość akumulatora i rejestru Rn, a wynik do A
- kod: 0 1 0 0 1 n2 n1 n0   n2...n0 - R0...R7   stąd: 48h-4Fh

ORL A,adres

- dodana zostaje logicznie zawartość A i komórki o podanym adresie „adres”, wynik do A
- kod: 01000101 45h

ORL A, @Ri
- dodana logicznie zostaje zawartość A i komórki o adresie podanym w Ri, wynik do A
- kod: 0100011i i=0,1 46h,47h

ORL A,#dana
- dodana zostaje logicznie zawartość A i stały argumentu (8-bitowa liczba), wynik do A
- kod: 01000100 44h

ORL adres, A
- dodana logicznie zostaje zawartość komórki a adresie „adres” i akumulatora, wynik do komórki o adresie „adres”
- kod: 01000010 42h

ORL adres, #dana
- dodana logicznie zostaje zawartość komórki o adresie „adres”, oraz argument stały (8-bitowa liczba), wynik do komórki o adresie „adres”
- kod: 01000011 43h

Instrukcjca XRL
- wykonywana jest suma mod 2 XOR wskazanych w instrukcji argumentów. Wynik operacji jest wpisywany do argumentu pierwszego instrukcji
- znaczniki: nie zmieniają się

XRL A, Rn
- zsumowana (mod 2) logicznie zostaje zawartość akumulatora i rejestru Rn, wynik w A
- kod: 0 1 1 0 1 n2 n1 n0 n2...n0 wskazują na R0...R7 stąd: 68h-6Fh

XRL A,adres
- zsumowana (mod 2) logicznie zostaje zawartość akumulatora i komórki o podanym adresie „adres”, wynik do A
- kod: 01100101 65h

XRL A, @Ri
- zsumowana (mod 2) logicznie zostaje zawartość akumulatora i komórki w wew. RAM o adresie wskazywanym przez rejestr Ri
- kod: 0110011i i=0,1 dtąd: 66h,67h

XRL A, #dana
- zsumowana (mod 2) logicznie zostaje zawartość akumulatora przez argument stały (8-bitowa liczba), wynik do A
- kod: 01100100 64h

XRL adres, A
- zsumowana (mod 2) logicznie zostaje zawartość komórki a adresie „adres” i akumulatora, wynik do komórki o adresie „adres”
- kod: 01100011 62h

XRL adres, #dana

- zsumowana (mod 2) logicznie zostaje zawartość komórki o adresie „adres” oraz argument stały (8-bitowa liczba), wynik do komórki o adresie „adres”
- kod: 01100011 63h

CLR A

- do akumulatora zostaje wpisana wartość 0 (wyzerowanie A)
- znaczniki: bez zmian
- kod: 11100100 E4h

CPL A

- wartość akumulatora zostaje zanegowana, wynik wpisany zostaje do akumulatora
- znaczniki: bez zmian
- kod: 11110100 F4h

RL A

- zawartość A zostaje przesunięta o 1 pozycję (1 bit) w lewo, to znaczy, że:
  bit 1 przyjmuje wartość bitu 0
  bit 2 przyjmuje wartość bitu 1
  itd...
  bit 7 przyjmuje wartość bitu 6
  a
  bit 0 przyjmuje wartość bitu 7
- znaczniki: bez zmian
- kod: 00100011 23h

RLC A

- zawartość A zostaje przesunięta w lewo o 1 pozycję (1 bit) z uwzględnieniem znacznika C, to znaczy że: znacznik C przyjmuje wartość bitu 7 (akumulatora oczywiście)
  bit 1 przyjmuje wartość bitu 0
  bit 2 przyjmuje wartość bitu 1
  itd...
  bit 7 przyjmuje wartość bitu 6
  znacznik C przyjmuje wartość bitu 7
  a
  bit 0 przyjmuje wartość znacznika C
- znaczniki: C jest ustawiony zgodnie z wynikiem operacji
- kod: 00110011 33h

RR A

- zawartość A zostaje przesunięta w prawo o 1 pozycję (0 1 bit), to znaczy, że:
  bit 1 przyjmuje wartość bitu 2
  itd...
  bit 6 przyjmuje wartość bitu 7
  a
  bit 7 przyjmuje wartość bitu 0
- znaczniki: bez zmian
- kod: 00000011 03h

RRC A

- zawartość A zostaje przesunięta o 1 pozycję (1 bit) w prawo z uwzględnieniem znacznika C, to znaczy, że:
  bit 0 przyjmuje wartość bitu 1
  bit 1 przyjmuje wartość bitu 2
  itd...
  bit 6 przyjmuje wartość bitu 7
  znacznik C przyjmuje wartość bitu 0
  a
  bit 7 przyjmuje wartość znacznika C
- znaczniki: C jest ustawiany zgodnie z wynikiem operacji
- kod: 00010011 13h

SWAP A

- wymieniona zostaje zawartość bitów 3...0 (mniej znaczący półbajt) i bitów 7...4 (bardziej znaczący półbajt akumulatora. Operacja ta jest równoważna 4-krotnemu przesunięciu zawartości akumulatora
- znaczniki: bez zmian
- kod: 11000100 C4h

JC rel

- sprawdzany jest bit przeniesienia C w słowie PSW (adres bajtu D0h), jeżeli jest ustawiony (C=1), to do licznika rozkazów PC jest dodawana 8-bitowa liczba „rel” (zapisana w kodzie U2), zostaje wykonany skok względny
  PC <- PC + 2, jeśli C=1, to PC <- PC + rel
- kod: 01000000 40h

<u>UWAGA!!!</u>

Jak widać wartość przesunięcia nie może przekroczyć liczb z zakresu U2, czyli od -128...127, innymi słowy skok względny może odbyć się tylko w pewnym „otoczeniu” („w górę” lub „w dół”) od instrukcji skoku. Jeżeli wartość przesunięcia jest ujemna, to skok nastąpi oczywiście w kierunku mniejszych adresów (do PC zostaje dodana liczba ujemna), w przeciwnym przypadku w kierunku adresów wzrastających. W przypadku umieszczenia etykiety (miejsca) skoku poza tym zakresem, prawidłowe obliczenie przesunięcia będzie niemożliwe, a w przypadku korzystania z kompilatora 8051 wystąpi komunikat o błędzie kompilacji mówiący o przekroczeniu zakresu skoku względnego. Programiście w takim przypadku nie pozostaje nic innego, jak poprawić błąd modyfikując program źródłowy.

JNC rel

- sprawdzany jest bit przeniesienia C w słowie PSW (adres bajtu: D0h), jeżeli jest wyzerowane (C=0), to do licznika rozkazów PC jest dodawana 8-bitowa liczba „rel” (zapisana w kodzie U2), zostaje wykonany skok względny PC <- PC+2, jeśli C=0, to PC <- PC + rel
- kod: 01010000 50h

JB bit, rel

- sprawdzany jest bit, którego adres podany jest w „bit”, jeżeli jest wstawiony (=1) to do licznika rozkazów PC jest dodawana 8-bitowa liczba „rel” (zapisana w kodzie U2), zostaje wykonany skok względny

PC <- PC + 3, jeśli (bit)=1, to PC <- PC + rel

- kod: 00100000 20h

JNB bit, rel

- sprawdzany jest bit, którego adres podany jest w „bit”, jeżeli jest wyzerowany (=0), to do licznika PC jest dodawana 8-bitowa liczba „rel” (zapisana w kodzie U2), zostaje wykonany skok względny

PC <- PC + 3, jeśli C=0, to PC <- PC + rel

- kod: 00110000 30h

JBC bit,rel

- sprawdzany jest bit, którego adres podany jest w „bit”, jeżeli jest ustawiony (=1), to zostaje wyzerowany, po czym do licznika PC jest dodawana 8-bitowa liczba „rel” (zapisana w kodzie U2), zostaje dokonany skok względny

PC <- PC + 3, jeśli (bit)=1, to (bit)<- 0 i PC <- PC + rel

- kod: 00010000 10h

LCALL adr16

- licznik rozkazów PC zostaje zwiększony o 3, jest ładowany na stos w efekcie czego wskaźnik stosu jest zwiększany o 2, a do licznika rozkazów PC zostaje wpisany 16-bitowy adres bezpośredni, podany jako argument przy wywołaniu instrukcji „LCALL”. Dzięki tej instrukcji możliwe jest wywołanie podprogramu w dowolnym obszarze pamięci (64KB) poprzez podanie bezpośredniego adresu
- kod: 00010010 12h

RET

- adres powrotu z podprogramu (zapamiętana wcześniej podczas instrukcji LCALL (lub ACALL - nie omawiana tutaj) wartość licznika PC) jest wpisywany do licznika rozkazów PC. Wskaźnik stosu zostaje pomniejszony o 2. W efekcie następuje powrót z podprogramu. Instrukcją tą musi się kończyć każdy podprogram (z wyjątkiem podprogramów obsługi przerwań, wywoływanych automatycznie przez procesor)
- kod: 00100010 22h

RETI

- adres powrotu z podprogramu obsługi przerwania jest wpisywany do licznika rozkazów PC. Wskaźnik stosu zostaje pomniejszony o 2. Instrukcją tą musi kończyć się każdy podprogram, który jest wywoływany automatycznie przez procesor w przypadku zgłoszenia przerwania o żądania jego obsługi
- kod: 00110010 32h

SJMP rel

- do zawartości licznika rozkazów PC jest dodawane 8-bitowe przesunięcie (liczba ze znakiem w kodzie U2 z zakresu <-128...127>). W efekcie wykonywany jest skok w obrębie 126 bajtów od kolejnej po SJMP instrukcji
- kod: 10000000 80h

JMP A+DPTR

- w wyniku tej operacji następuje skok pod adres będący sumą aktualnej wartości rejestru DPTR (liczba 16-bitowa)i wartości akumulatora (liczba 8-bitowa). Można powiedzieć, że skok następuje pod adres w pamięci programu umieszczony w DPTR z przesunięciem podanym w akumulatorze. Przesunięcie to traktowane jest jako liczba bez znaku, czyli z zakresu <0...255>
- kod: 01110011 73h

JZ rel

- sprawdzana jest zawartość akumulatora, jeżeli jest równa zero, to do licznika rozkazów PC dodawane jest przesunięcie „rel" (liczba 8-bitowa ze znakiem w kodzie U2)
- kod: 01100000 60h

JNZ rel

- sprawdzana jest zawartość A, jeżeli jest różna od zera, to do licznika rozkazów PC dodawane jest przesunięcie „rel"
- kod: 01110000 70h

Instrukcja ***CJNE <arg1> , <arg2> , rel***

- w instrukcji tej porównywane są dwa argumenty: arg1 i arg2. Jeżeli nie są one równe, to do zawartości licznika rozkazów PC jest dodawane przesunięcie „rel" na zasadach zgodnych z opisywanymi wcześniej. W efekcie wykonany zostaje skok w programie. Tu UWAGA, skok następuje względem instrukcji występującej po instrukcji CJNE. **Dodatkowo jest zmieniany znacznik przeniesienia C. I tak, jeżeli w wyniku porównania okaże się, że argument arg1 jest mniejszy od argumentu arg2 to do znacznika C wpisywana jest jedynka (znacznik ustawiony), w przeciwnym wypadku znacznik jest zerowany (C=0).** W zależności od typu argumentów instrukcji możliwe są cztery przypadki, oto one.

CJNE A, adres, rel

- porównywany jest A oraz komórka wew. RAM o adresie „adres"
- kod: 10110101 B5h

CJNE A, #dana, rel

- akumulator zostaje porównany z argumentem bezpośrednim (8-bitowa liczba), jeżeli nie są zgodne ot następuje skok

- kod: 10110100   B4h

CJNE Rn, #dana, rel

- rejestr Rn (R0...R7) zostaje porównany z argumentem bezpośrednim, jeżeli nie są zgodne zostaje wykonany skok
- kod: 1 0 1 1 0 n2 n1 n0   n2...n0 określają jeden z rejestrów R0...R7 stąd: B8-BFh

CJNE @Ri, #dana, rel

- porównana zostaje zawartość komórki w wew. RAM której adres znajduje się w rejestrze Ri z argumentem bezpośrednim. Jeżeli się różnią, to następuje skok
- kod: 1011011i   i=0,1   stąd: B6, B7

Instrukcja   ***DJNZ   <arg>, rel***

- w wyniku tej operacji od wskazanego argumentu „arg” odejmowana jest jedynka (jest on dekrementowany). Jeżeli w wyniku odjęcia wartość argumentu nie jest równa zero, to zostaje wykonany skok zgodnie z zasadami opisywanymi wcześniej w przypadku argumentu „rel”. Stan znaczników nie zmienia się. W zależności od typu argumentu rozróżnia się dwa typy instrukcji, oto one

DJNZ Rn, rel

- zmniejszona zostaje zawartość podanego rejestru Rn (R0...R7) o jeden, a następnie jeżeli nie jest równa zero, to następuje skok
- kod: 1 1 0 1 1 n2 n1 n0   n2...n0 wskazuje na R0...R7 stąd: D8-DFh

DJNZ adres, rel

- zmniejszona zostaje zawartość komórki pamięci wew. RAM o podanym bezpośrednio adresie o jeden, a następnie jeżeli nie jest równa zero, to następuje skok
- kod: 11010101   D5h

NOP

- nie rób nic
- w wyniku tej instrukcji nie zmienia się stan procesora, z wyjątkiem licznika rozkazów który po pobraniu tej instrukcji jest zwiększany o jeden
- kod: 00000000   00h